XenSummit Asia
November 2-3, 2011
Seoul, Korea

Hardware accelerated Virtualization in the ARM Cortex™ Processors

John Goodacre
Director, Program Management
ARM Processor Division

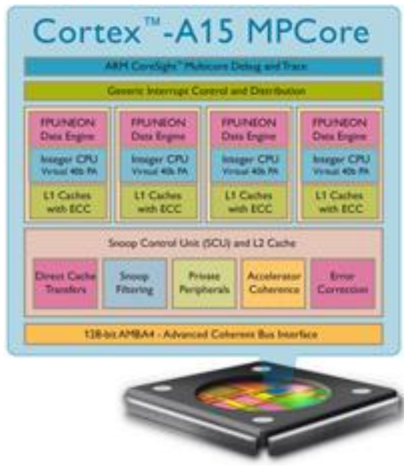
ARM Ltd. Cambridge UK
2nd November 2010

Sponsored by:

SAMSUNG & KOREA UNIVERSITY & kt

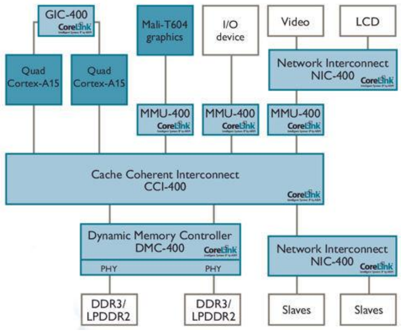
New Capabilities in the Cortex-A15

- Full compatibility with the Cortex-A9
  - Supporting the ARMv7 Architecture


Cortex™-A15 MPCore
ARM CoreSight™ Multicore Debug and Trace
Generic Interrupt Control and Distribution
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
Snoop Control Unit (SCU) and L2 Cache
Direct Cache Transfers
Snoop Filtering
Private Peripherals
Accelerator Coherence
Error Correction
128-bit AMBA4 - Advanced Coherent Bus Interface

- Addition of Virtualization Extension (VE)
  - Run multiple OS binary instances simultaneously
  - Isolates multiple work environments and data

- Supporting Large Physical Addressing Extensions (LPAE)
  - Ability to use up to 1TB of physical memory



- With AMBA 4 System Coherency (AMBA-ACE)
  - Other cached devices can be coherent with processor
  - Many core multiprocessor scalability
  - Basis of concurrent big.LITTLE Processing

Large Physical Addressing

- Cortex-A15 introduces 40-bit physical addressing
  - Virtual memory (apps and OS) still has 32bit address space

- Offering up to 1 TB of physical address space
  - Traditional 32bit ARM devices limited to 4GB

- What does this mean for ARM based systems?
  - Reduced address-map congestion
  - More applications at the same time
  - Multiple resident virtualized operating systems
  - Common global physical address in many-core

Virtualization Extensions: The Basics

- New Non-secure level of privilege to hold Hypervisor
  - Hyp mode
- New mechanisms avoid the need Hypervisor intervention for:
  - Guest OS Interrupt masking bits
  - Guest OS page table management
  - Guest OS Device Drivers due to Hypervisor memory relocation
  - Guest OS communication with the interrupt controller (GIC)
- New traps into Hyp mode for:
  - ID register accesses and idling (WFI/WFE)
  - Miscellaneous “difficult” System Control Register cases
- New mechanisms to improve:
  - Guest OS Load/Store emulation by the Hypervisor
  - Emulation of trapped instructions through syndromes

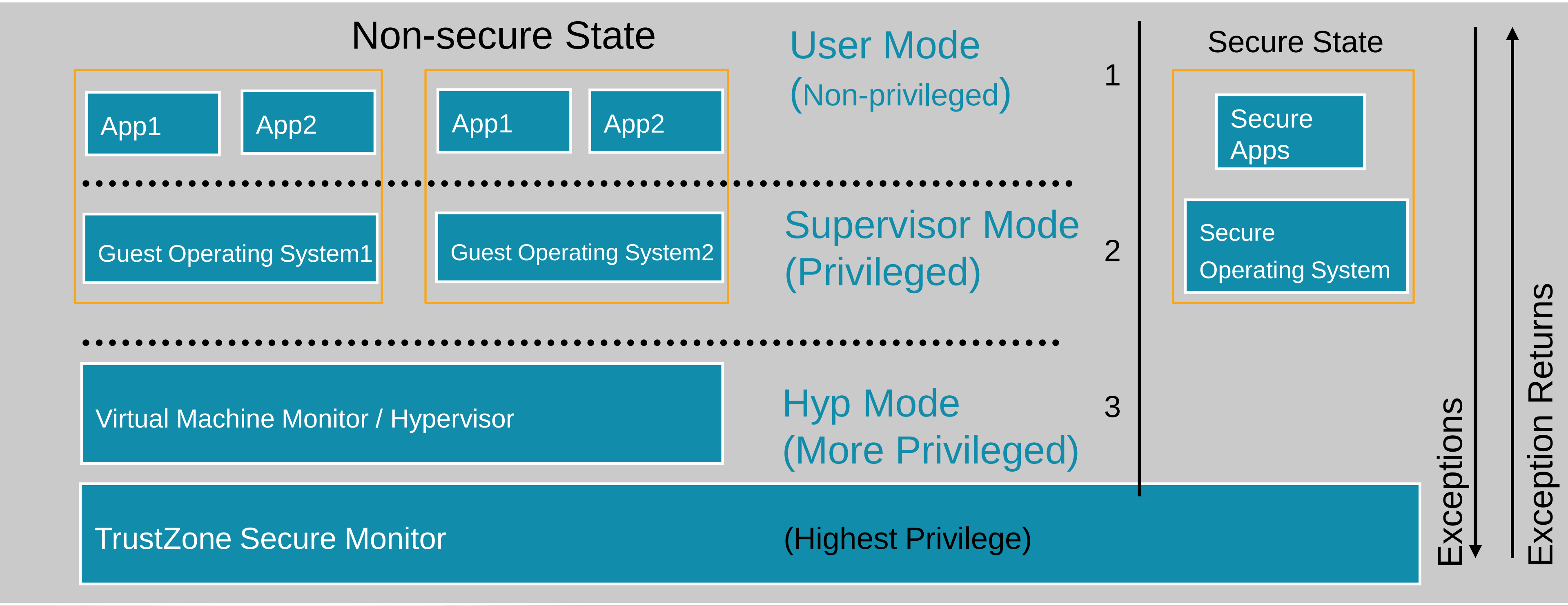
How does ARM do Virtualization

- Extensions to the v7-A Architecture, available on the Cortex™-A15 and Cortex-A7 CPUs
  - Second stage of address translation (separate page tables) Functionality for virtualizing interrupts inside the Interrupt Controller
  - Functionality for virtualizing all CPU features, including CP15
  - Option of a MMU within the system to help virtualize IO

- Hypervisor runs in new "Hyp" exception mode / privilege
  - HVC (Hypervisor Call) instruction to enter Hyp mode
  - Uses previously unused entry (0X14 offset) in vector table for hypervisor traps
  - Hyp mode exception link register, SPSR, stack pointer
  - Hypervisor Control Register (HCR) marks virtualized resources
  - Hypervisor Syndrome Register (HSR) for Hyp mode entry reason

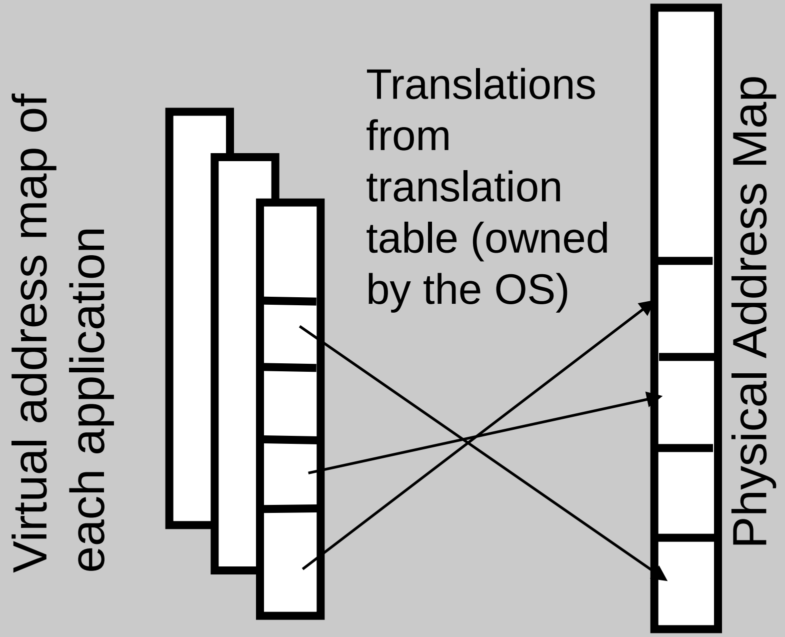
Virtualization: Third Privilege

- Guest OS same kernel/user privilege structure
- HYP mode higher privilege than OS kernel level
- VMM controls wide range of OS accesses
- Hardware maintains TZ security (4th privilege)

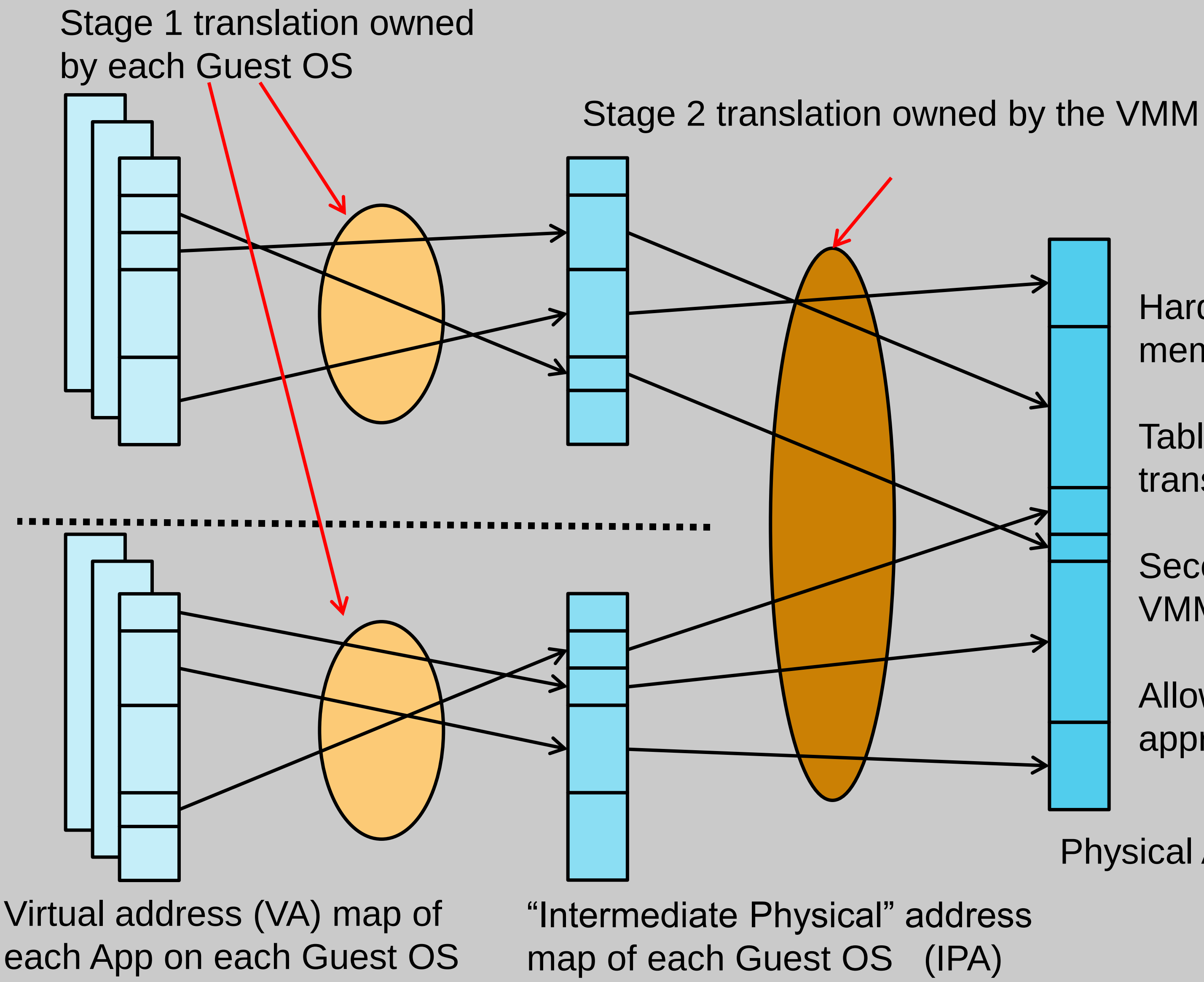


Memory – the Classic Resource

- Before virtualisation – the OS owns the memory
  - Allocates areas of memory to the different applications
  - Virtual Memory commonly used in “rich” operating systems


Virtual address map of each application
Translations from translation table (owned by the OS)
Physical Address Map

Virtual Memory in Two Stages


Stage 1 translation owned by each Guest OS
Stage 2 translation owned by the VMM
Virtual address (VA) map of each App on each Guest OS
“Intermediate Physical” address map of each Guest OS (IPA)

Hardware has 2-stage memory translation

Tables from Guest OS translate VA to IPA

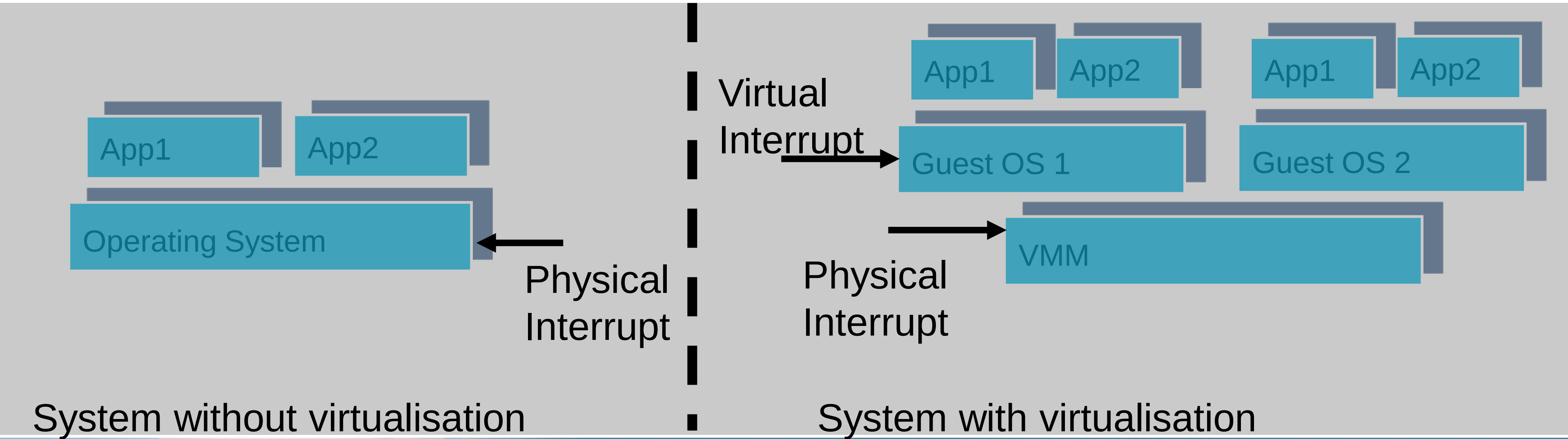
Second set of tables from VMM translate IPA to PA

Allows aborts to be routed to appropriate software layer

Physical Address (PA) Map

Classic Issue: Interrupts

- An Interrupt might need to be routed to one of:
  - Current or different GuestOS
  - Hypervisor
  - OS/RTOS running in the secure TrustZone environment
- Basic model of the ARM virtualisation extensions:
  - Physical interrupts are taken initially in the Hypervisor
  - If the Interrupt should go to a GuestOS :
    - Hypervisor maps a “virtual” interrupt for that GuestOS



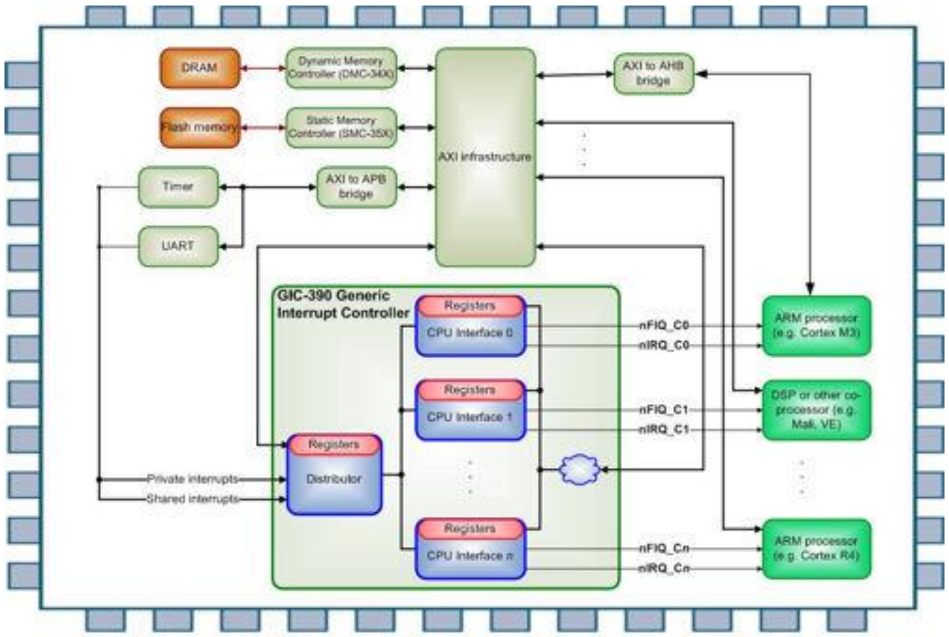
Interrupt Virtualization

- Virtualisation Extensions provides :
  - Registers to hold the Virtual Interrupt
  - CPSR.{I,A,F} bits in the GuestOS only appling to that OS
    - Physical Interrupts are not masked by the CPSR.{I.A.F} bits
    - GuestOS changes to I,A,F no longer need to be trapped
  - Mechanism to route all physical interrupts to Monitor Mode
    - Already utilized in TrustZone technology based devices
  - Virtual Interrupts are routed to the Non-secure IRQ/FIQ/Abort
  - Guest OS manipulates a virtualized interrupt controller
- Actually available in the Cortex-A9 to aid paravirtualization support for interrupts

Virtual Interrupt Controller

- New “Virtual” GIC Interface has been Architected
  - ISR of GuestOS interacts with the virtual controller
  - Pending and Active interrupt lists for each GuestOS
  - Interacts with the physical GIC in hardware
  - Creates Virtual Interrupts only when priority indicates it is necessary

- GuestOS ISRs therefore do not need calls for:
  - Determining interrupt to take [Read of the Interrupt Acknowledge]
  - Marking the end of an interrupt [Sending EOI]
  - Changing CPU Interrupt Priority Mask [Current Priority]



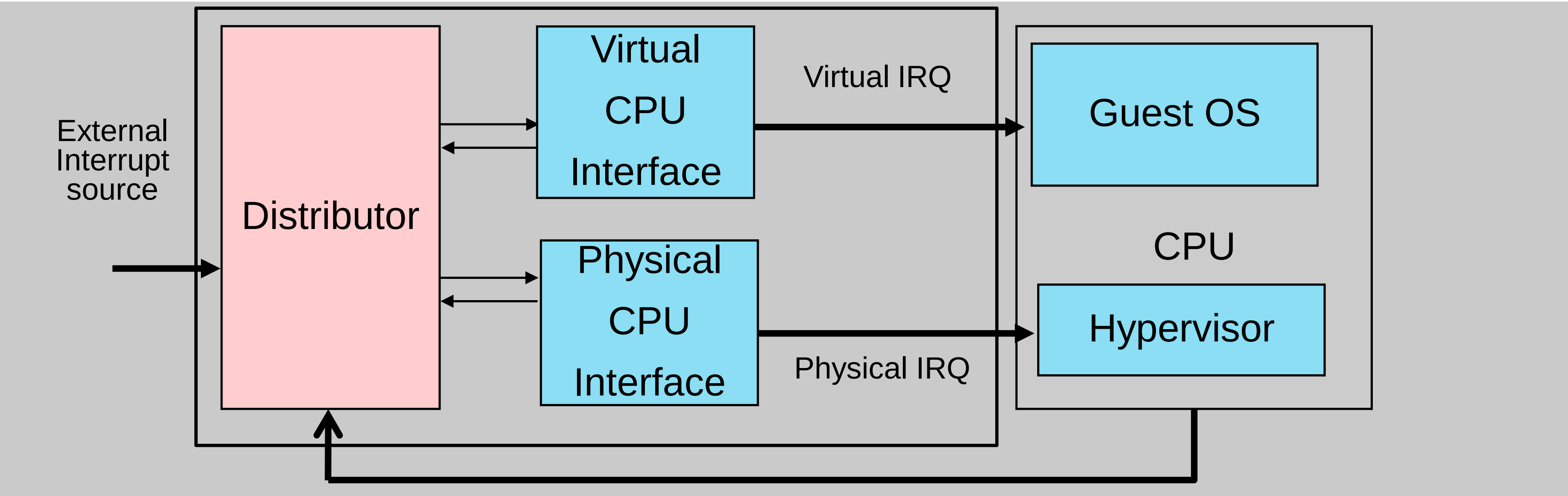
GIC-390 Generic Interrupt Controller in Multiprocessor System

Virtual GIC

- GIC now has separate sets of internal registers:
  - Physical registers and virtual registers
  - Non-virtualized system and hypervisor access the physical registers
  - Virtual machines access the virtual registers
  - Guest OS functionality does not change when accessing the vGIC
- Virtual registers are remapped by hypervisor so that the Guest OS thinks it is accessing the physical registers
  - GIC registers and functionality are identical
- Hypervisor can set IRQs as virtual in the HCR
  - Interrupts are configured to generate a Hypervisor trap
  - Hypervisor can deliver an interrupt to a CPU running a virtual process using “register lists” of interrupts

Virtual interrupt example

- External IRQ (configured as virtual by the hypervisor) arrives at the GIC
- GIC Distributor signals a Physical IRQ to the CPU
- CPU takes HYP trap, and Hypervisor reads the interrupt status from the Physical CPU Interface
- Hypervisor makes an entry in register list in the GIC
- GIC Distributor signals a Virtual IRQ to the CPU
- CPU takes an IRQ exception, and Guest OS running on the virtual machine reads the interrupt status from the Virtual CPU Interface



Resource Ownership

- Software-only approaches
  - Access to resources by GuestOS intercepted by the VMM
    - VMM interprets the GuestOS's intent
    - Provides its own mechanism to meet that intent
  - Mechanism of interception varies
    - Paravirtualisation adds a hypercall to the source code
    - Binary translation adds a hypercall to the binary
    - Exceptions in Hardware provide an trapping of operations
  - Hypercalls can be more efficient
    - More of the intent to be expressed in a single VMM entry

- Hardware assisted approaches:
  - Provide further indirection to resources
  - Accelerating trapped operations by syndrome information

Helping with Virtual Devices

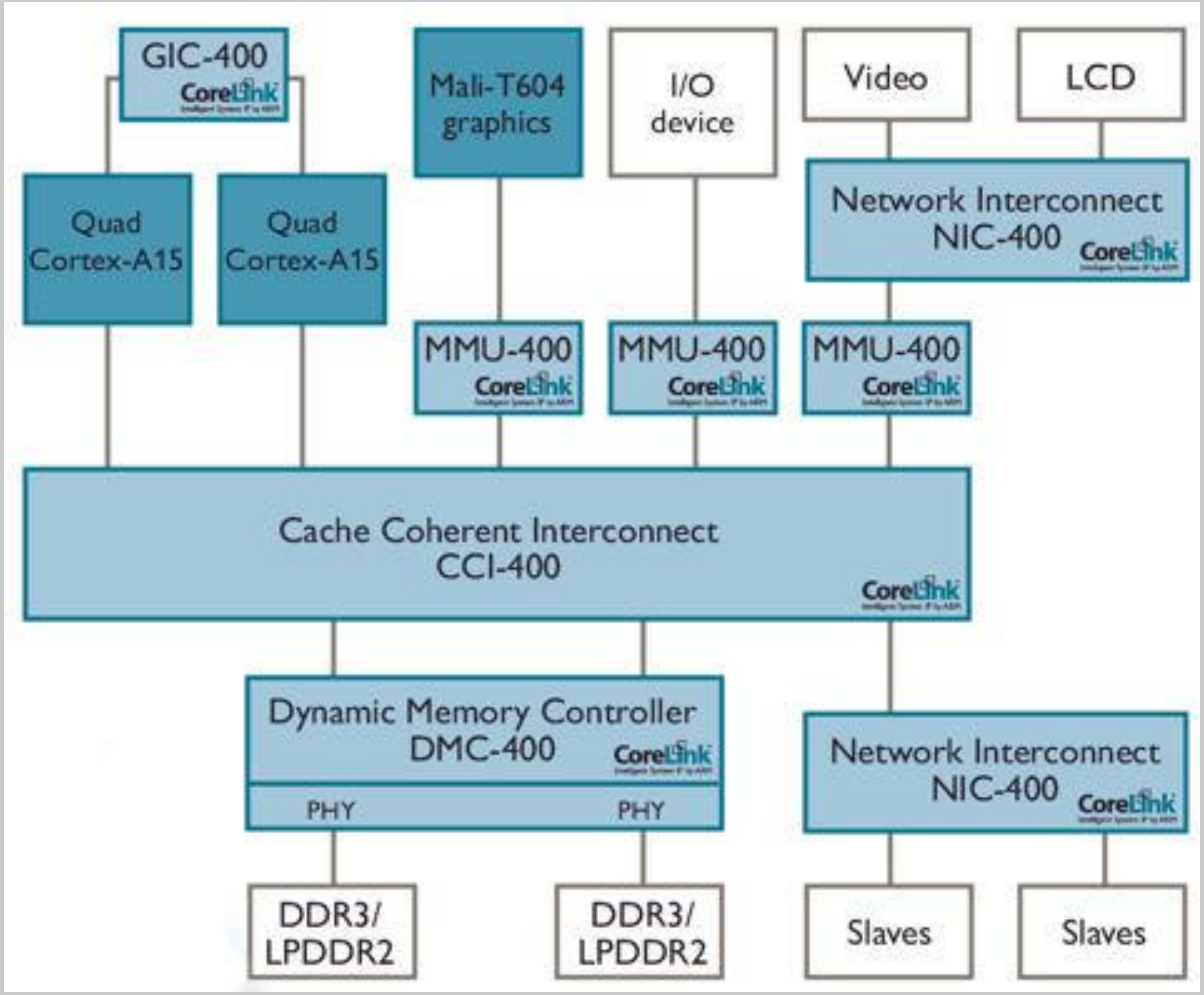
- ARM I/O handling uses memory mapped devices
  - Reads and Writes to the Device registers have specific side-effects
- Creating “Virtual Devices” requires emulation:
  - Typically reads/writes to devices have to trap to the VMM
  - VMM interprets the operation and performs emulation
  - Perfect virtualization means all possible devices loads/stores emulated
  - Fetching and interpreting emulated load/store is performance intensive

  - “Syndrome” information on aborts available for some loads/stores
  - Syndrome unpacks key information about the instruction
    - Source/Destination register, Size of data transfer, Size of the instruction, SignExtension etc
    - If syndrome not available, then fetching of the instruction for emulation still required

Devices and Memory

- Providing address translation for devices is important
  - Allows unmodified device drivers in the GuestOS
  - If the device can access memory, GuestOS will program it in IPA
- ARM virtualisation adds option for a “System MMU”
  - Enables second stage memory translations in the system
- A System MMU could also provide stage 1 translations
  - Allows devices to be programmed into guest’s VA space
- System MMU natural fit for the processor ACP port
  - ARM defining a common programming model
  - Intent is for the system MMU to be hardware managed using Distributed Virtual Messages found in AMBA 4 ACE

Potential of System MMU



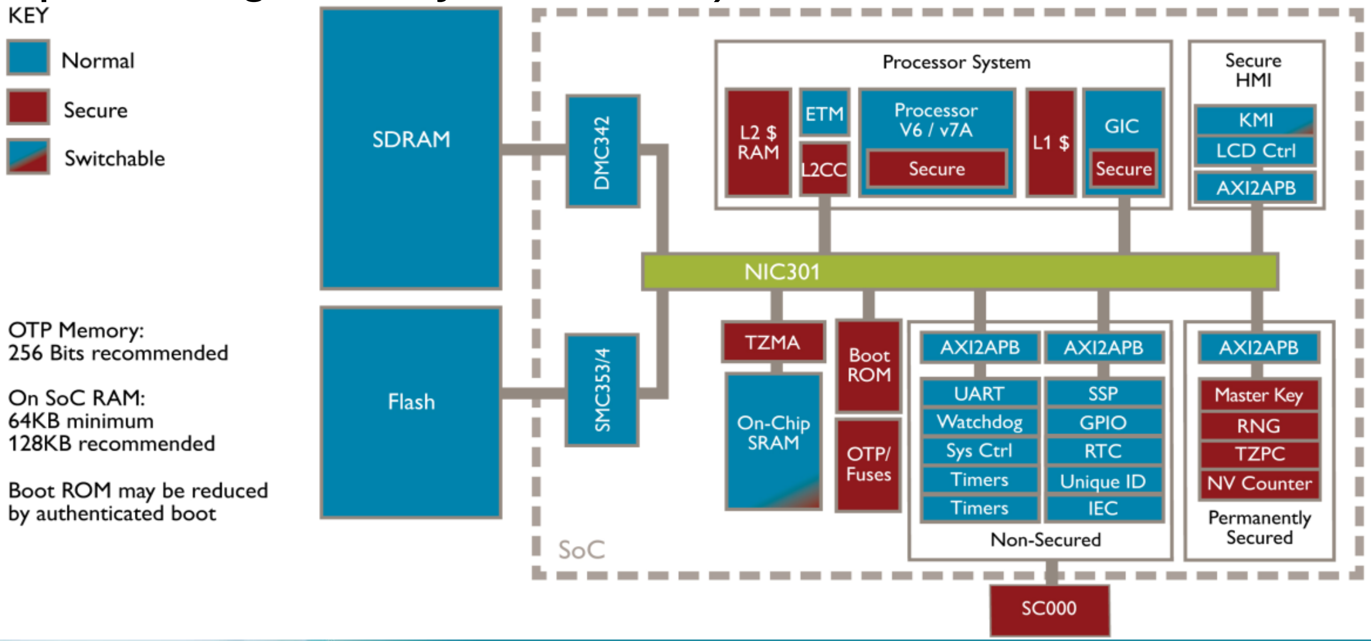
Partitioning in a Secure ARM System

TrustZone®
Security Foundation by ARM®

- ARM TrustZone technology define two worlds
  - Everything must live in Normal World or Secure World
- TrustZone-Enhanced processor exports World Information
  - Via NS bit (Not Secure) on system bus since AMBA 3 AXI
- TrustZone-Aware devices can partition across both worlds
  - Only AMBA AXI compatible devices can be TrustZone-aware
- AMBA 3 AXI Interconnect decodes TZ like an address line
- AMBA AHB and APB do not contain TrustZone information
  - AHB and APB devices live in only one World
  - Groups of peripherals can be managed from bus interface
  - Inclusion of TrustZone Peripheral Controller gives more control

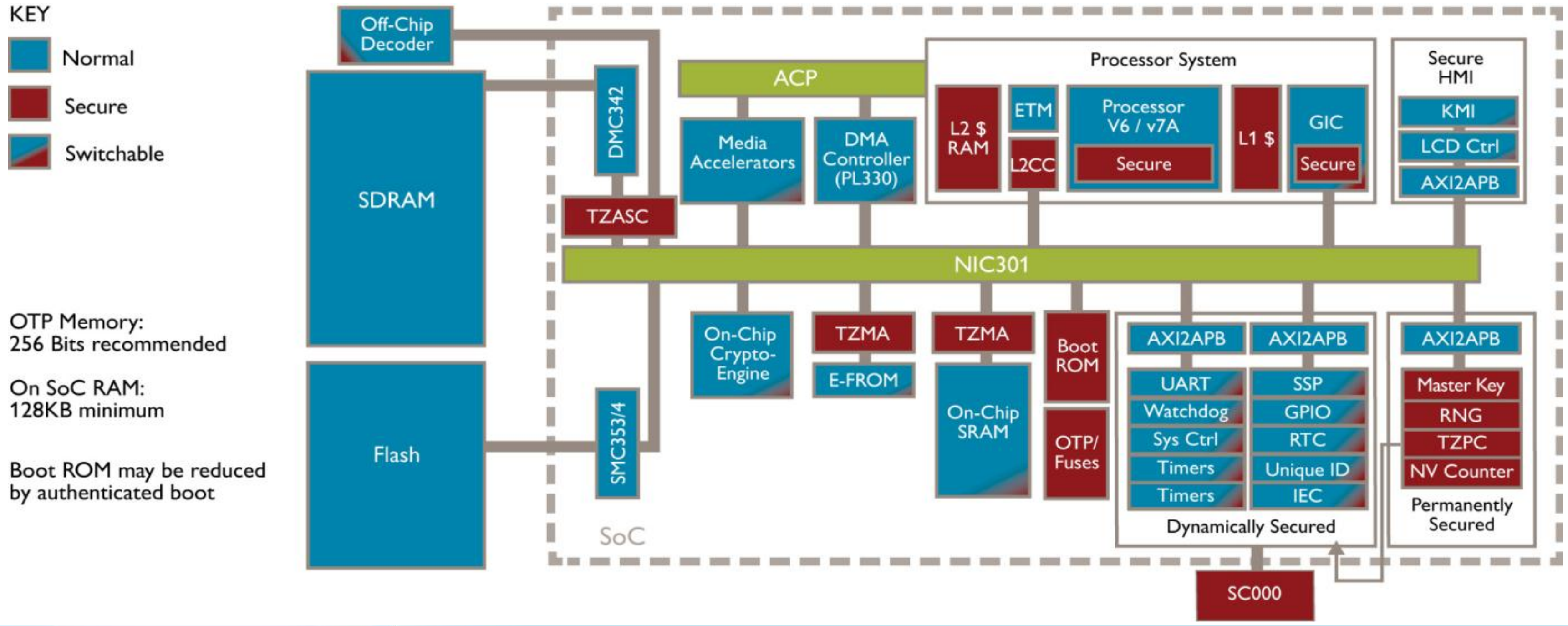
Base Secure System

- Minimal TrustZone System required for payment solutions
  - Protects On-Chip Secure Ram area via TrustZone Memory Adaptor
  - Keyboard and screen secured dynamically to protect PIN entry
  - Master Key and Random Number Generators for daughter-keys permanently secures. Non-volatile counters required for state management & anti-rollback (fuse based not non-volatile memory due to process geometry limitations)

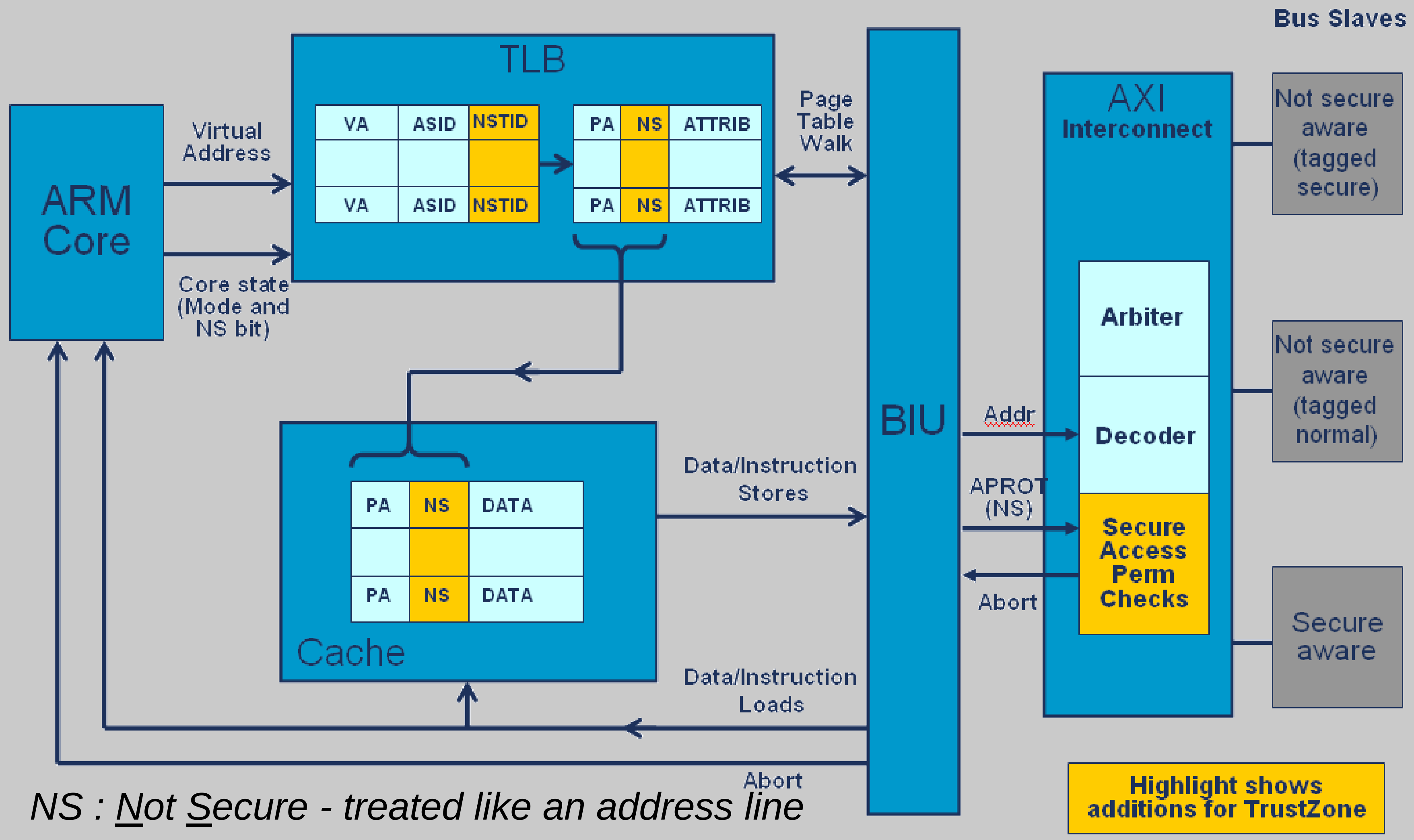


Extended Secure System

- Extended TrustZone system enables complex content management
  - Builds on Base Secure System
  - + TrustZone ASC to protect media in RAM and off-chip decode
  - + On-chip Crypto, Media Accelerators & DMA Controller for media handling



Propagating System Security



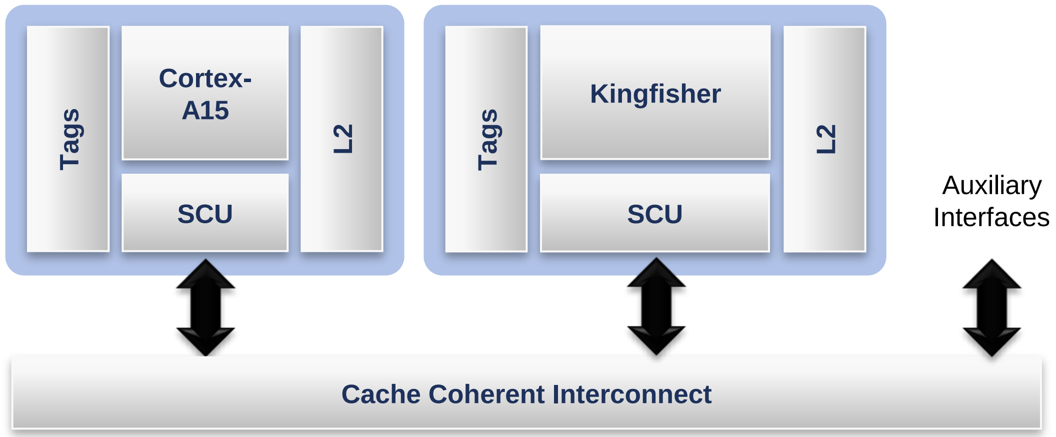
NS : Not Secure - treated like an address line

Multi-Cluster Virtualization

- Works just like with a single cluster MPCore system
  - Guest OS (like threads) can migrate from CPU to CPU across clusters
- External (virtual) GIC used to handle interrupts
  - Functions the same as internal GIC, but accessed by multiple CPUs
- AMBA Coherency Extensions (ACE)
  - Manages coherency across clusters
- System MMU allows other bus masters to map from IPA to PA
- Hypervisor needs to be aware of different clusters and CPUs
  - But again: it is just like a single cluster system
  - Hardware requirements are the same as non-virtualized multi-cluster
    - Just make sure there's enough memory

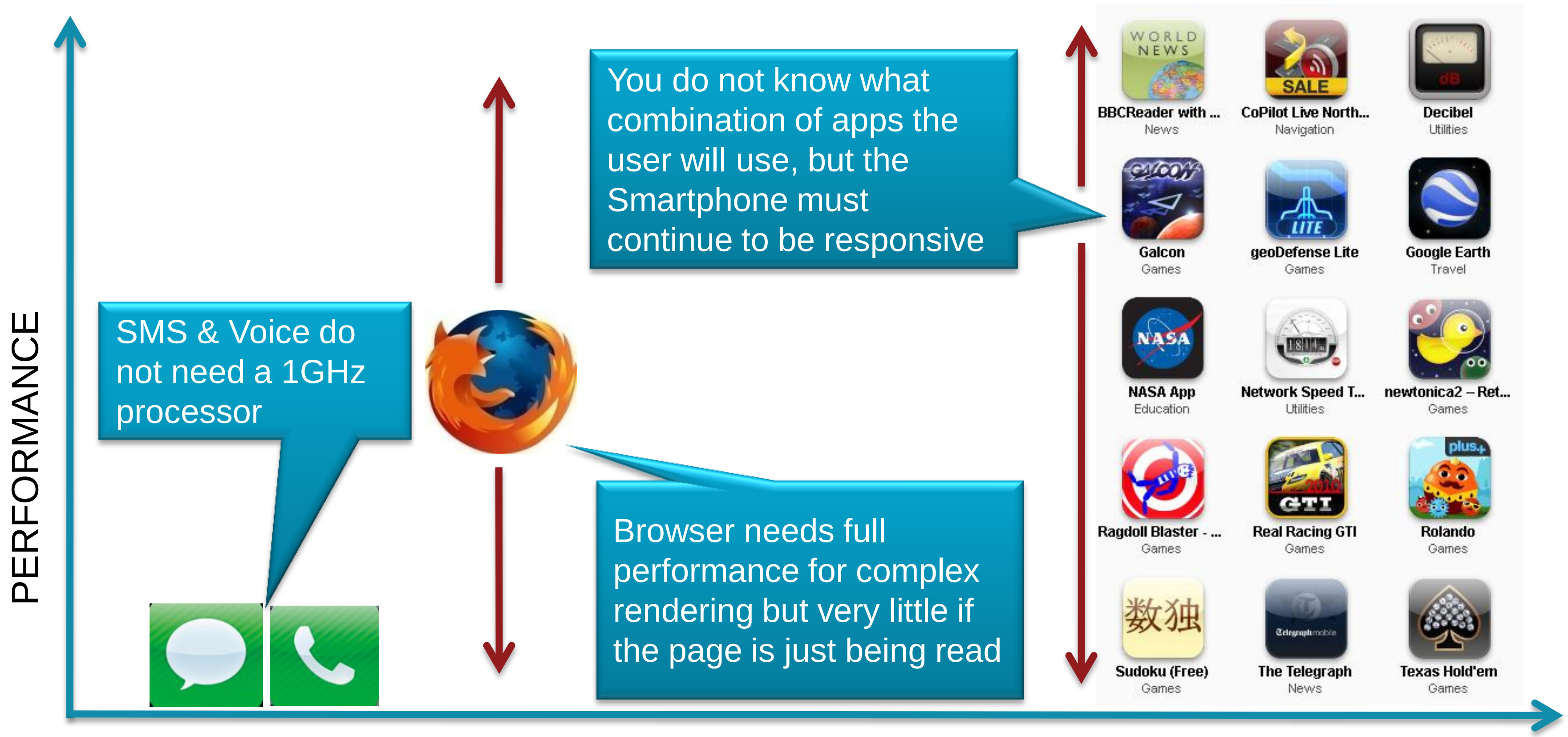
big.LITTLE Multi-Processing

- "Big" processor is paired with a "little" processor
  - Processors share exact same ISA and feature set
  - High performance tasks run on the Big processor
  - Lightweight/non-time-critical tasks run on the little processor
  - Best of both worlds solution for high performance and low power



- big.LITTLE Use Models
  - big.LITTLE Switch (Swapping) – one CPU cluster active at a time
  - big.LITTLE MP – both CPUs can be active, loads dynamically balanced

System Characteristics of big.LITTLE

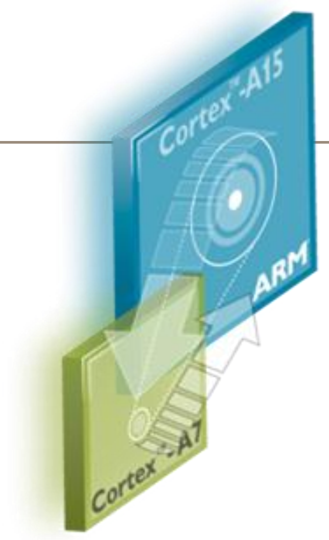


SMS & Voice do not need a 1GHz processor

Browser needs full performance for complex rendering but very little if the page is just being read

You do not know what combination of apps the user will use, but the Smartphone must continue to be responsive

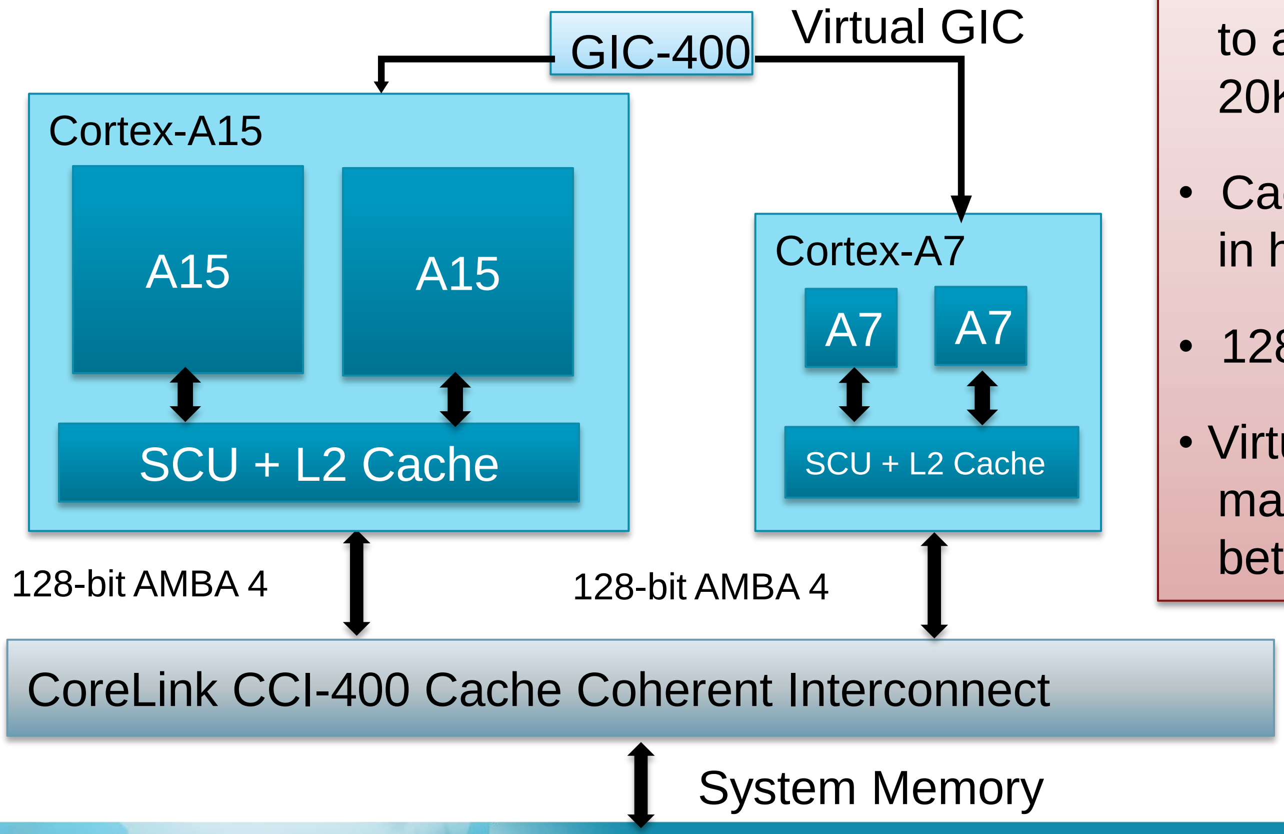
big.LITTLE Processing



- Right sized Core for the Right Task
  - Cortex-A7 enabled by default with sufficient performance for common usage scenarios
  - Cortex-A15 performance for best user experience
- Software alignment allows execution to migrate between cores
  - Transparent to user, application and OS
- Optimizes system workload based on application requirements
  - Extending existing power management
  - Additional benefit though OS Power Management Policy tuning
- End-product delivers longer battery life and richer user experience at the same time

Putting together a big.LITTLE System

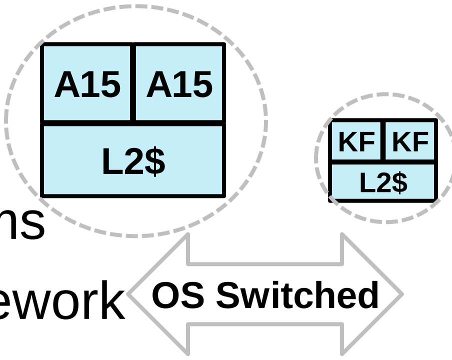
- Cortex-A15 and Cortex-A7 clusters are cache coherent
- CCI-400 maintains cache-coherency between clusters
- GIC-400 provides transparent virtualized Interrupt control
- Supports all big.LITTLE usage models



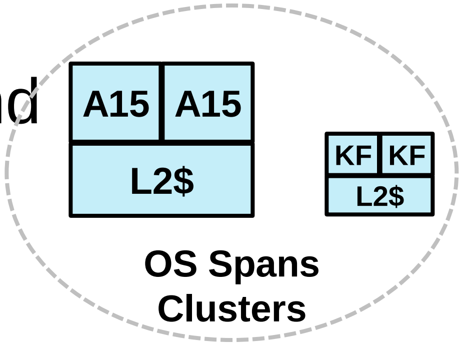
- Full task migration (active to active) in less than 20K cycles
- Cache coherency managed in hardware
- 128-bit system transactions
- Virtualization manager can map OS either across or between clusters

Forms of big.LITTLE

- big.LITTLE cluster switching
  - OS sees big cores or little cores at any one time
  - Eases deployment of software on big.LITTLE platforms
    - Minimizes OS changes by extending DVFS framework
  - Symmetric big/little clusters currently preferred

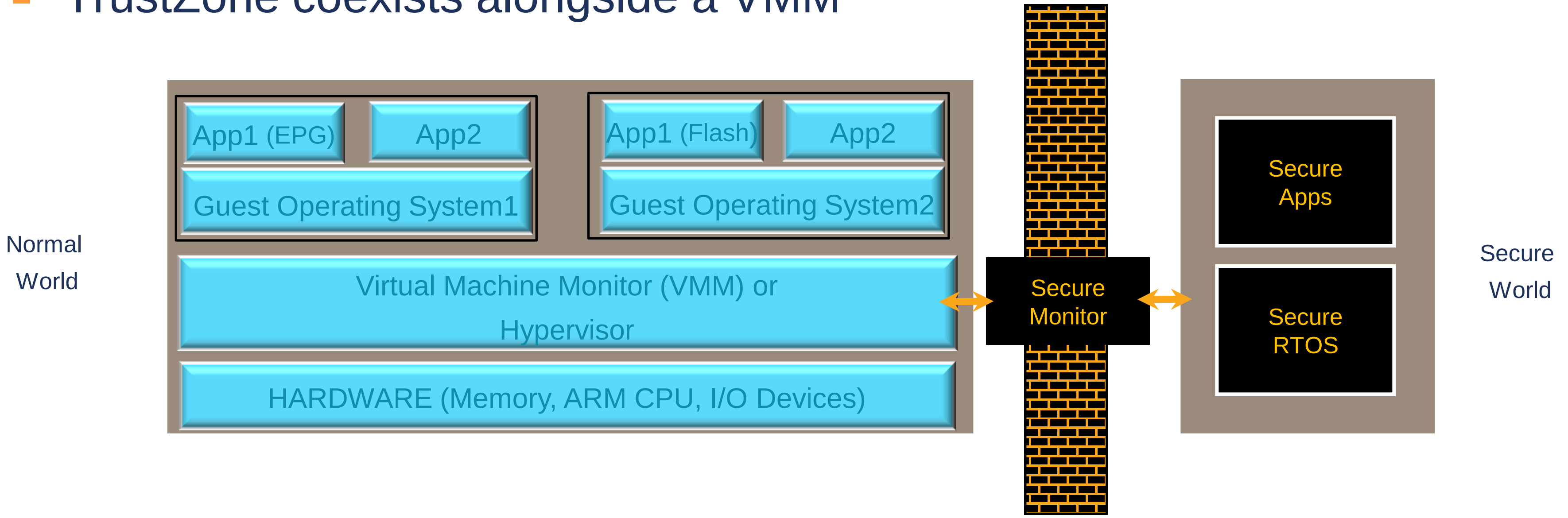


- Concurrent big.LITTLE
- OS sees all CPUs all the time; better energy and performance matching of compute to workload
  - Asymmetric clusters supported
  - Expected to be the preferred use model eventually
  - Requires OS improvements and changes for efficient operation
    - Co-ordination of MP scheduling, power policies, environmental conditions



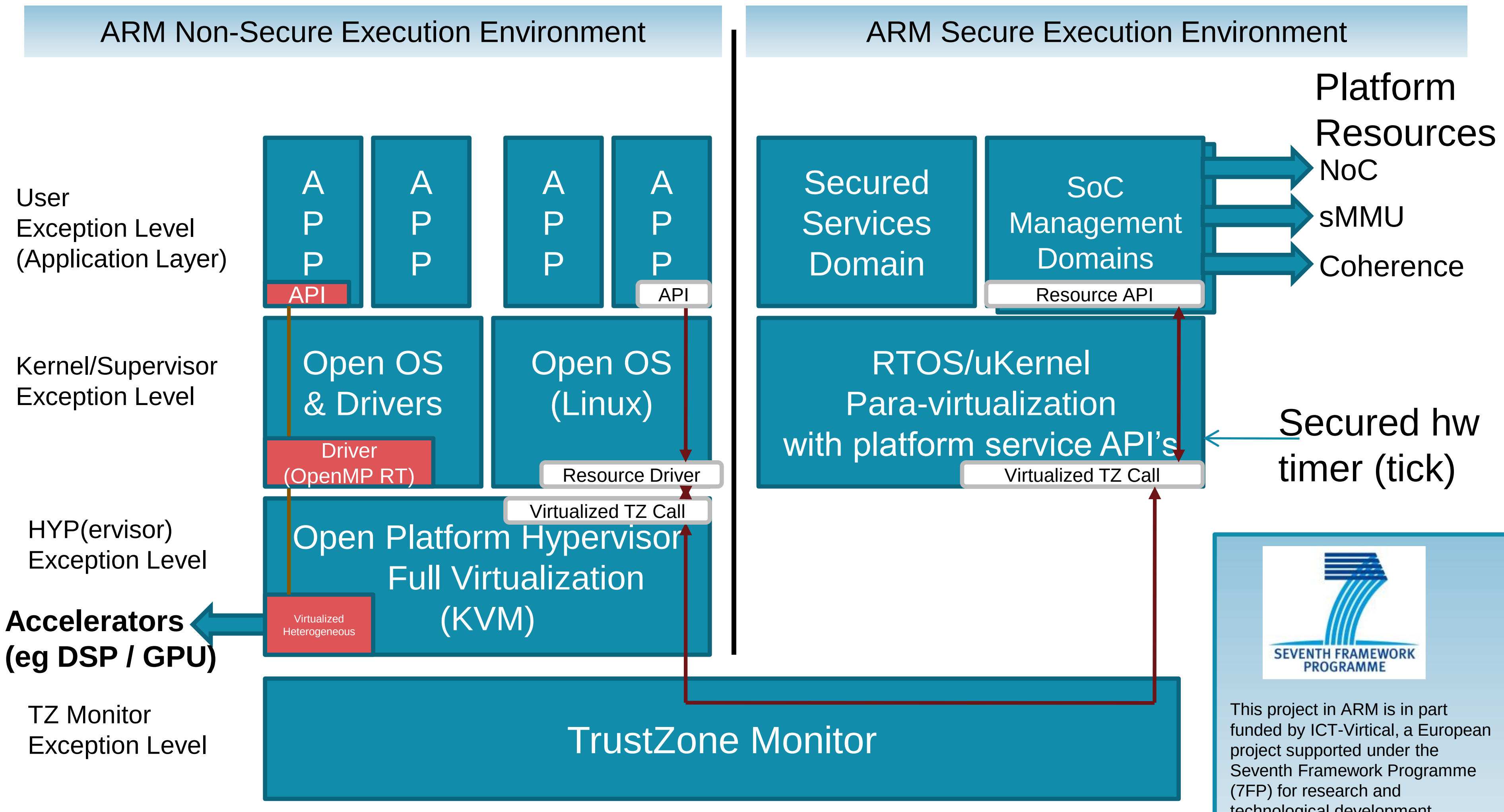
Virtualization and TrustZone

- TrustZone coexists alongside a VMM



- TrustZone offers a specialised type of Virtualisation
  - Only 2 ‘Worlds’ – not extendable (except through paravirtualization)
    - Although VMM can also span both worlds
  - Fourth privilege level is provided by CPU’s secure monitor mode
  - Non-symmetrical - The two ‘Worlds’ are not equal
    - Secure world can access both worlds (33bit addressing)

Spanning Hypervisor Framework


ARM Non-Secure Execution Environment
ARM Secure Execution Environment
Platform Resources
NoC
sMMU
Coherence
User Exception Level (Application Layer)
A P P
A P P
A P P
A P P
API
API
Secured Services Domain
SoC Management Domains
Resource API
Kernel/Supervisor Exception Level
Open OS & Drivers
Driver (OpenMP RT)
Open OS (Linux)
Resource Driver
RTOS/uKernel Para-virtualization with platform service API's
Virtualized TZ Call
Secured hw timer (tick)
HYP(ervisor) Exception Level
Virtualized TZ Call
Open Platform Hypervisor Full Virtualization (KVM)
Virtualized Heterogeneous
Accelerators (eg DSP / GPU)
TZ Monitor Exception Level
TrustZone Monitor

SEVENTH FRAMEWORK PROGRAMME

This project in ARM is in part funded by ICT-Virtical, a European project supported under the Seventh Framework Programme (7FP) for research and technological development

Project website
http://www.virtical.eu

Summary

- Hardware based Virtualization Extensions were introduced in the soon-available Cortex-A15
  - Also found the in the recently announce Cortex-A7
- Enables full binary compatible virtualization of the CPU
  - Supporting both existing and new market scenarios
- Extended into the system and IO through the system MMU
- Independent of the TrustZone environment
  - Providing both a secure environment, and a virtualizable non-secured environment
- Software models available now, first silicon 2H2012